

## METHOD AND APPARATUS FOR DATA REDUNDANCY ELIMINATION AT THE BLOCK LEVEL

### BACKGROUND OF THE INVENTION

5

#### 1. Technical Field:

The present invention relates to data processing and, in particular, to size reduction of files or objects in a data processing system. Still more particularly, the present invention provides a method, apparatus, and program for data redundancy elimination at the block level.

#### 2. Description of Related Art:

Despite increasing capacities of storage systems and network links, there are often benefits to reducing the size of file objects that are stored and/or transmitted. Examples of environments that would see such benefits include mobile devices with limited storage, communication over telephone links, or storage of reference data, which is data that is written, saved permanently, and often never again accessed. Other examples include wide-area transfers of large objects, such as scientific data sets, or over saturated links. The present invention is concerned with self-contained storage systems, in which all data is stored in a single location. Data can take the form of files in a file system, objects in a database, or other storage, and the terms "object," "file," and "file object" are used interchangeably in this document.

Numerous techniques for reducing large object sizes exist including data compression, duplicate suppression, and delta encoding. Data compression is the elimination of redundancy internally within an object. Duplicate suppression is the process of eliminating redundancy caused by identical objects. Delta encoding eliminates

redundancy of an object relative to another object, which may be an earlier version of the object having the same name.

Another technique involves a method for dividing larger objects into smaller, variable-sized “chunks” and eliminating duplicate chunks. The boundaries of the chunks may be determined, for example, using a function called a Rabin fingerprint over a sliding window of the content. The Rabin fingerprint is only one such solution and other techniques may be used to efficiently and deterministically hash the content. Such content-defined blocks isolate changes within an object, so that changes in one part of an object do not affect other parts and duplication of blocks of content across objects can be detected. This technique was first proposed for the low-bandwidth file system (LBFS) and has since been applied to other systems.

**Figure 1** illustrates a process of dividing objects into blocks and duplicate blocks. File object **110** is divided into a plurality of content-defined blocks or “chunks” **120**. The process then compares the chunks and removes exact matches, replacing each duplicate chunk with a reference to an identical chunk **132** in the plurality of chunks **130**. By replacing duplicate chunks with references to identical chunks, the size of the file object is reduced. However, the LBFS technique requires chunks to be identical. If one byte is different between two chunks, no benefit is realized between those two chunks. Thus, an object that has many minor changes scattered throughout may see no improvements whatsoever from the LBFS technique.

Yet another technique is known as delta encoding via resemblance detection (DERD). This technique attempts to extend delta encoding by identifying similar objects that may otherwise have no association, either spatial or temporal, with the object being encoded. The technique then performs delta encoding of the object against a chosen similar object. The resemblance detection step typically uses Rabin fingerprints to compute a set of values based on the contents of the object and then deterministically

select a small number of these values to represent each object. Two objects with many of these fingerprints in common are likely to have much of their content in common overall.

**Figure 2** illustrates a process of delta encoding via resemblance detection. File objects **202** and **204** share common content. The process performs resemblance detection and delta encodes object **204** relative to object **202**. The process results in a delta encoded file object **214**. By replacing objects with delta encoded objects, a reduction in object size is realized. However, the performance of the DERD technique does not scale well with large datasets, because the resemblance detection step uses a quadratic algorithm which can suffer a performance penalty with large datasets. DERD also does not detect multiple objects when the objects match different parts of the encoded object. For example, if file A consists of the concatenation of files B-Z, then each file B-Z closely resembles a portion of file A; however, DERD is not capable of detecting the resemblance, because the fingerprints of each file B-Z would intersect only a small portion of the fingerprints of file A.

Another technique is an optimization of the Rsync protocol. Rsync allows two versions of a file to be synchronized across a slow link by sending hashes of blocks of content and identifying when one copy has the same blocks as the other, possibly offset between the two copies. A multi-round version of Rsync has been devised, which tries large blocks and then decomposes them into smaller blocks to find pieces that are similar enough to delta encode.

## SUMMARY OF THE INVENTION

The present invention recognizes the disadvantages of the prior art and provides a redundancy elimination mechanism, which applies aspects of duplicate block elimination and delta encoding at the block level. The redundancy elimination mechanism of the present invention realizes benefits in both spatial reduction and performance. The redundancy elimination mechanism divides file objects into blocks or “chunks.” These chunks may be content-defined blocks or fixed-length blocks. Identical chunks are suppressed. The redundancy elimination mechanism also performs resemblance detection on remaining chunks to identify chunks with sufficient redundancy to benefit from delta encoding of individual chunks. Any remaining chunks that do not benefit from delta encoding are compressed. Resemblance detection is optimized by merging groups of fingerprints into super fingerprints. This merging can be constructed to ensure that if two objects have a single super fingerprint in common, they are extremely likely to be within a specified threshold of similarity.

## BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use,  
5 further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

**Figure 1** illustrates a process of dividing objects into blocks and duplicate blocks;

**Figure 2** illustrates a process of delta encoding via resemblance detection;

10 **Figure 3** depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented;

**Figure 4** is a block diagram of a data processing system that may be implemented as a server in accordance with a preferred embodiment of the present invention;

15 **Figure 5** is a block diagram of a data processing system in which the present invention may be implemented;

**Figures 6A-6C** are block diagrams illustrating the operation of a redundancy elimination mechanism in accordance with a preferred embodiment of the present invention;

20 **Figures 7A and 7B** are block diagrams illustrating the computation of fingerprints for chunks in accordance with a preferred embodiment of the present invention;

**Figure 8** is a diagram illustrating a data redundancy elimination and compression mechanism in accordance with a preferred embodiment of the present invention; and

25 **Figures 9A and 9B** are flowcharts illustrating the operation of a redundancy elimination mechanism in accordance with a preferred embodiment of the present invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, **Figure 3** depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented.

5 Network data processing system **300** is a network of computers in which the present invention may be implemented. Network data processing system **300** contains a network **302**, which is the medium used to provide communications links between various devices and computers connected together within network data processing system **300**. Network **302** may include connections, such as wire, wireless communication links, or fiber optic  
10 cables.

In the depicted example, server **304** is connected to network **302** and provides access to storage unit **306**. Storage unit **306** may be a self-contained system, such as, for example, a hard drive, a backup system, a storage area network (SAN), etc. In addition, clients **308**, **310**, and **312** are connected to network **302**. These clients **308**, **310**, and **312**  
15 may be, for example, personal computers or network computers. In the depicted example, server **304** provides data, such as boot files, operating system images, and applications to clients **308-312**. Clients **308**, **310**, and **312** are clients to server **304**. Network data processing system **300** may include additional servers, clients, and other devices not shown.

20 In accordance with a preferred embodiment of the present invention, devices in network data processing system **300** include a mechanism for redundancy elimination at the block level (REBL). For example, server **304** may store reference data, which is written and permanently saved but may not be accessed again. In this case, server **304** may benefit by reducing the size of file objects that are stored in storage unit **306**. As  
25 another example, one of clients **308-312** may have limited storage capacity. As a specific example, a client may be a mobile computing device with a small hard drive or flash

memory for permanent storage. In such an instance, the client device may benefit from reduced object size.

The redundancy elimination mechanism of the present invention applies aspects of duplicate block elimination and delta encoding at the block level. The redundancy  
 5 elimination mechanism divides file objects into blocks or “chunks.” The chunks may be content-defined blocks or fixed-length blocks. Identical chunks are suppressed. The redundancy elimination mechanism also performs resemblance detection on remaining chunks to identify chunks with sufficient redundancy to benefit from delta encoding of individual chunks. Any chunks that do not benefit from delta encoding are compressed.  
 10 Resemblance detection is optimized by merging groups of fingerprints into super fingerprints. This merging can be constructed to ensure that if two objects have a single super fingerprint in common, they are extremely likely to be within a specified threshold of similarity. Objects are substantially similar if they are similar enough to benefit from delta encoding.

15 In the depicted example, network data processing system **300** is the Internet with network **302** representing a worldwide collection of networks and gateways that use the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands  
 20 of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system **300** also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). **Figure 3** is intended as an example and not as an architectural limitation for the present invention.

25 Referring to **Figure 4**, a block diagram of a data processing system that may be implemented as a server, such as server **304** in **Figure 3**, is depicted in accordance with a preferred embodiment of the present invention. Data processing system **400** may be a

symmetric multiprocessor (SMP) system including a plurality of processors **402** and **404** connected to system bus **406**. Alternatively, a single processor system may be employed. Also connected to system bus **406** is memory controller/cache **408**, which provides an interface to local memory **409**. I/O bus bridge **410** is connected to system bus **406** and  
 5 provides an interface to I/O bus **412**. Memory controller/cache **408** and I/O bus bridge **410** may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **414** connected to I/O bus **412** provides an interface to PCI local bus **416**. A number of modems may be connected to PCI local bus **416**. Typical PCI bus implementations will support four PCI expansion slots or  
 10 add-in connectors. Communications links to clients **308-312** in **Figure 3** may be provided through modem **418** and network adapter **420** connected to PCI local bus **416** through add-in connectors.

Additional PCI bus bridges **422** and **424** provide interfaces for additional PCI local buses **426** and **428**, from which additional modems or network adapters may be supported.  
 15 In this manner, data processing system **400** allows connections to multiple network computers. A memory-mapped graphics adapter **430** and hard disk **432** may also be connected to I/O bus **412** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 4** may vary. For example, other peripheral devices, such as optical disk drives and  
 20 the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

The data processing system depicted in **Figure 4** may be, for example, an IBM eServer pSeries system, a product of International Business Machines Corporation in  
 25 Armonk, New York, running the Advanced Interactive Executive (AIX) operating system or Linux operating system. “eServer” and “pSeries” are trademarks of International Business Machines Corporation.



With reference now to **Figure 5**, a block diagram of a data processing system is shown in which the present invention may be implemented. Data processing system **500** is an example of a computer, such as client **308** in **Figure 3**, in which code or instructions implementing the processes of the present invention may be located. In the depicted

5 example, data processing system **500** employs a hub architecture including a north bridge and memory controller hub (MCH) **508** and a south bridge and input/output (I/O) controller hub (ICH) **510**. Processor **502**, main memory **504**, and graphics processor **518** are connected to MCH **508**. Graphics processor **518** may be connected to the MCH through an accelerated graphics port (AGP), for example.

10 In the depicted example, local area network (LAN) adapter **512**, audio adapter **516**, keyboard and mouse adapter **520**, modem **522**, read only memory (ROM) **524**, hard disk drive (HDD) **526**, CD-ROM driver **530**, universal serial bus (USB) ports and other communications ports **532**, and PCI/PCIe devices **534** may be connected to ICH **510**. PCI/PCIe devices may include, for example, Ethernet adapters, add-in cards, PC cards for

15 notebook computers, etc. PCI uses a cardbus controller, while PCIe does not. ROM **524** may be, for example, a flash binary input/output system (BIOS). Hard disk drive **526** and CD-ROM drive **530** may use, for example, an integrated drive electronics (IDE) or serial advanced technology attachment (SATA) interface. A super I/O (SIO) device **536** may be connected to ICH **510**.

20 An operating system runs on processor **502** and is used to coordinate and provide control of various components within data processing system **500** in **Figure 5**. The operating system may be a commercially available operating system such as Windows XP, which is available from Microsoft Corporation. "Windows XP" is a trademark of Microsoft Corporation. An object oriented programming system such as Java may run in conjunction

25 with the operating system and provides calls to the operating system from Java programs or applications executing on data processing system **500**. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented programming

system, and applications or programs are located on storage devices, such as hard disk drive 526, and may be loaded into main memory 504 for execution by processor 502. The processes of the present invention are performed by processor 502 using computer implemented instructions, which may be located in a memory such as, for example, main memory 504, memory 524, or in one or more peripheral devices 526 and 530.

Those of ordinary skill in the art will appreciate that the hardware in **Figure 5** may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash memory, equivalent non-volatile memory, or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 5**.

Also, the processes of the present invention may be applied to a multiprocessor data processing system.

**Figures 6A-6C** are block diagrams illustrating the operation of a redundancy elimination mechanism in accordance with a preferred embodiment of the present invention. More particularly, with reference to **Figure 6A**, file object 610 is divided into “chunks” 620. These chunks are variable-sized and content defined. The redundancy elimination mechanism uses a method such as Rabin fingerprints to determine chunk boundaries in an object. A Rabin fingerprint of a data chunk is computed by performing a polynomial modulo of the chunk with a predetermined irreducible polynomial. A Rabin fingerprint is computed for overlapping portions of the file. When the fingerprint has certain characteristics, such as its low-order bits equaling a constant value, the bytes producing that fingerprint constitute the end of a chunk, called a break-point. Rabin fingerprints are efficient for computing on a moving window in a file, thereby making it easy to incrementally scan and divide the object into chunks.

The redundancy elimination mechanism of the present invention then compares the chunks and removes exact matches. Chunks that are identical to other chunks are replaced with a reference to the identical chunk 632 in the resulting set of chunks 630. For instance, in the example shown in **Figure 6A**, the reference to identical chunk 632

may refer to chunk **634** or another chunk not shown in the illustrated example. Note also that fixed-sized chunks or blocks may be used instead of variable-sized content defined blocks. In some systems, fixed-sized blocks may result in comparable benefits with somewhat lower overhead; however, in most cases, fixed-sized blocks are expected to detract from the benefits of redundancy elimination.

With reference now to **Figure 6B**, the redundancy elimination mechanism performs resemblance detection on the remaining chunks in the set of chunks **630**. Rabin fingerprints are normally computed at the time chunks are defined. These Rabin fingerprints may be used to identify content features in each chunk. Thus, each chunk may have a plurality of associated fingerprints. Resemblance detection determines the number of matching features in the chunks. This predetermined number may be received from an operator, a configuration file, a program constant, or other ways. Resemblance detection may consist of counting the number of matching fingerprints between objects. If two chunks have a predetermined number of matching fingerprints, the chunks are determined to be substantially similar or, in other words, similar enough to benefit from delta encoding. However, other techniques for identifying substantially similar chunks may also be used within the scope of the present invention.

Delta encoding consists of storing the differences of one chunk relative to a chosen similar chunk, called a reference chunk. Substantially similar chunks are replaced in the set of chunks **640** with a delta encoded chunk **642**. Since a delta encoded chunk stores only the differences between similar chunks, the size of a delta encoded chunk will be significantly less than the original chunk. In the example depicted in **Figure 6B**, the delta encoded chunk **642** may be encoded relative to chunk **644** or another chunk not shown in the illustrated example.

Turning to **Figure 6C**, the redundancy elimination mechanism compresses any remaining chunks in the set of chunks **640**. Any chunks that are not identical to other chunks and do not benefit from delta encoding are compressed using traditional

techniques. In the example shown in **Figure 6C**, compressed chunk **652** results in the set of chunks **650**. Therefore, the redundancy elimination mechanism of the present invention realizes the benefits of duplicate block suppression, resemblance detection and delta encoding, and traditional compression techniques.

5           **Figures 7A and 7B** are block diagrams illustrating the computation of fingerprints for chunks in accordance with a preferred embodiment of the present invention. More particularly, with reference to **Figure 7A**, an object is divided into a set of chunks **720**. A plurality of fingerprints **730** is computed for the chunks with each chunk having a plurality of fingerprints corresponding to features of the chunk. Resemblance detection  
10 may consist of counting the number of matching fingerprints between objects. If two chunks have a predetermined number of matching fingerprints, the chunks are determined to be substantially similar or, in other words, similar enough to benefit from delta encoding.

            With reference now to **Figure 7B**, an object is divided into a set of chunks **720**.  
15 A plurality of fingerprints **730** is computed for the chunks with each chunk having a plurality of fingerprints corresponding to features of the chunk. The plurality of fingerprints for a chunk are merged together into a fingerprint group, such as group **732**. A super fingerprint is then calculated for each fingerprint group, resulting in super fingerprints **740**. An exemplary embodiment of the super fingerprint calculation is to  
20 concatenate each fingerprint and compute a hash of the concatenated fingerprints, for instance with an MD5 hash, producing a fixed-size 128-bit super fingerprint. Each chunk may then have one or more super fingerprints.

            There are a number of details that are important to the efficient and effective use of the redundancy elimination technique of the present invention. For example, the size  
25 of a chunk must be optimized. Smaller chunks increase the rate of similar chunks being identified. However, smaller chunks increase the overhead both in storing the fingerprints and in performing resemblance detection computations. Smaller chunks also

decrease the possible benefit of simple compression. Furthermore, a method for determining a reference block for delta encoding must be decided upon. For example, a reference block may be selected when it matches the greatest number of matching features of any candidate reference block. Alternatively, the first reference block to  
 5 match N super fingerprints of another block can be used as its reference block.

In addition, the number of fingerprints to be clustered into a super fingerprint must be determined. If there are many fingerprints incorporated into each super fingerprint, then if two objects have a single super fingerprint in common, they are very likely to be substantially similar. The fewer fingerprints there are per super fingerprint,  
 10 the less accurate the resemblance detection, but having too many fingerprints in each super fingerprint can lead to identifying fewer similar blocks and lower overall savings. By having fewer fingerprints per super fingerprint, the likely amount of similarity decreases. Therefore, a balance between accuracy (effectiveness) and performance (efficiency) must be resolved.

**Figure 8** is a diagram illustrating a data redundancy elimination and compression mechanism in accordance with a preferred embodiment of the present invention. File object **802** is provided to data redundancy elimination and compression mechanism **810**. The data redundancy elimination and compression mechanism includes fingerprint module **812**, which calculates fingerprints that are used to divide the object into chunks  
 20 and to define features within each chunk. Fingerprint module **812** may also generate super fingerprints for groups of fingerprints.

Block division module **814** divides the object into blocks or “chunks.” The object may be divided into fixed-sized blocks or variable-sized blocks. Dividing the object into fixed-sized blocks may avoid a performance penalty, particularly at the time of division.  
 25 Fixed-sized blocks may also suffer a performance penalty when determining resemblance, however. Variable-sized content defined blocks or “chunks” may result in a higher likelihood of identical and/or substantially similar chunks being detected. Block

division module **814** may divide object **802** into variable-sized content defined chunks using fingerprints generated by fingerprint module **812**.

Block comparison and elimination module **816** compares blocks to identify identical blocks. Blocks that are identical to other blocks are replaced with a reference to another reference block. Resemblance detection module **818** performs resemblance detection computations to identify blocks that are substantially similar or, in other words, similar enough to benefit from delta encoding. The resemblance detection module may, for example, count the number of fingerprints that blocks have in common. Alternatively, the resemblance detection module determines whether blocks have matching super fingerprints. Blocks that have a predetermined number of features in common are determined to be substantially similar. Delta encoding module **820** performs delta encoding on a chunk, storing differences between the chunk and a reference chunk. These delta encoded chunk replaces the original chunk, thus reducing the storage or transmission size. The delta encoding module may increase the number of blocks that are delta-encoded using heuristics, such as first-fit versus best-fit, to increase the overall effectiveness.

The compression module **822** performs compression on any remaining chunks. The compression module may use any known compression techniques to compress the size of individual blocks. The resulting object, reduced object **830**, is a set of references to identical blocks, delta encoded blocks, and compressed blocks. In an exemplary embodiment, the compression module **822** may compress the object **802**. If the compressed object has a smaller size than reduced object **830**, the compressed object may be used for storage or transmission.

**Figures 9A and 9B** are flowcharts illustrating the operation of a redundancy elimination mechanism in accordance with a preferred embodiment of the present invention. More particularly, with reference to **Figure 9A**, a process for performing redundancy elimination at the block level is illustrated. The process begins and divides

the object into blocks (step 902). As stated above, the blocks may be fixed-sized blocks or content defined chunks. Then, the process identifies duplicate blocks (step 904) and eliminates duplicate blocks (step 906).

5        Thereafter, the process identifies similar blocks (step 908). The process then performs delta encoding on similar blocks (step 910) to form delta encoded blocks. The process compresses the remaining blocks that do not benefit from delta encoding (step 912) and ends.

10        Next, with reference to **Figure 9B**, a process for determining whether to compress an object is illustrated. The process begins and compresses the object (step 952). The process then compares the size of the compressed object with the results of the reduced results from the reduction process shown in **Figure 9A** (step 954). A determination is made as to whether the compressed object is smaller in size than the reduced results (step 956). If the compressed object is smaller, the process uses the compressed object (step 958) and ends. However, if the object reduced by the reduction technique is smaller in  
15        step 956, the process uses the object resulting from the reduction process shown in **Figure 9A** (step 960) and ends.

20        The process in **Figure 9B** determines whether to use the compressed object based on the size of the compressed object relative to the results of the reduction technique shown in **Figure 9A**. However, other measures of effectiveness may also be used. For example, whether compression will result in a greater speed of execution may be used as a measure of effectiveness.

25        Thus, the present invention solves the disadvantages of the prior art by providing a technique for redundancy elimination at the block level. The redundancy elimination mechanism of the present invention applies aspects of several techniques to attain benefits in both spatial reduction and performance. The redundancy elimination mechanism divides file objects into fixed-sized blocks or content-defined blocks or “chunks.” Identical chunks are suppressed. The redundancy elimination mechanism also

performs resemblance detection on remaining chunks to identify chunks with sufficient redundancy to benefit from delta encoding of individual chunks. Any remaining chunks that do not benefit from delta encoding are compressed. Resemblance detection is optimized by merging groups of fingerprints into super fingerprints. If two objects have  
5 a specific number of super fingerprints in common, they are extremely likely to be substantially similar.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed  
10 in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, DVD-ROMs, and transmission-type media, such as digital and analog  
15 communications links, wired or wireless communications links using transmission forms, such as, for example, radio frequency and light wave transmissions. The computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

The description of the present invention has been presented for purposes of  
20 illustration and description and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various  
25 modifications as are suited to the particular use contemplated.